Introduction to Machine Learning and Classical Supervised Learning

John Armstrong, Astronomy & Astrophysics

University of Glasgow

Science & Technology Facilities Council

ScotDIST

What is Machine Learning?

- Machine learning is the process of using statistical techniques to give computers the ability to learn how to perform a specific task *without* being explicitly programmed

    1. We use machine learning algorithms to approximate statistical models that may be too complex to work out

    2. This is done using a data-driven model approach: more data implies better model

Data-driven Models

4	13	6	11	8
A	B	A	B	A

7
?

Data-driven Models

4	13	6	11	8
A	B	A	B	A

7
?

Consistent Hypotheses:

- odd implies B
- less than 10 implies A

University of Glasgow

Data-driven Models

4	13	6	11	8
A	B	A	B	A

7
?

Consistent Hypotheses:

- odd implies B
- less than 10 implies A

Not enough data to get unambiguous solutions

University of Glasgow

Data-driven Models

4	13	6	11	8	2
A	B	A	B	A	B

7
?

University of Glasgow

Data-driven Models

4	13	6	11	8	2
A	B	A	B	A	B

7
?

Only consistent hypothesis:

- prime implies B

Data-driven Models

4	13	6	11	8	2
A	B	A	B	A	B

7
?

Only consistent hypothesis:

- prime implies B

Space of viable models shrinks with number of examples

Learning from Data

- As shown, our data can highly influence the model we use to describe it
- With an exponential growth we cannot feasibly look through all of our data to look for the optimal model to use
- This is why we turn to automation and machine learning

Learning from Data

- Given correct prior assumptions about our data and what we’re looking for we should be able to automate the process of finding the most optimised model

- This can be more challenging than just applying a pre-built algorithm due to data complexity

- There are a variety of ways to get to our data driven model

University of Glasgow

Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

University of Glasgow

Supervised Learning I

- This is when you know the structure of you data and want to create a model that will predict a characteristic of this data for an unknown point
- As such, these methods are good for coping with two types of problems: classification and regression
- Understanding whether your machine learning task is classification or regression is the key for selecting the right algorithms to use.

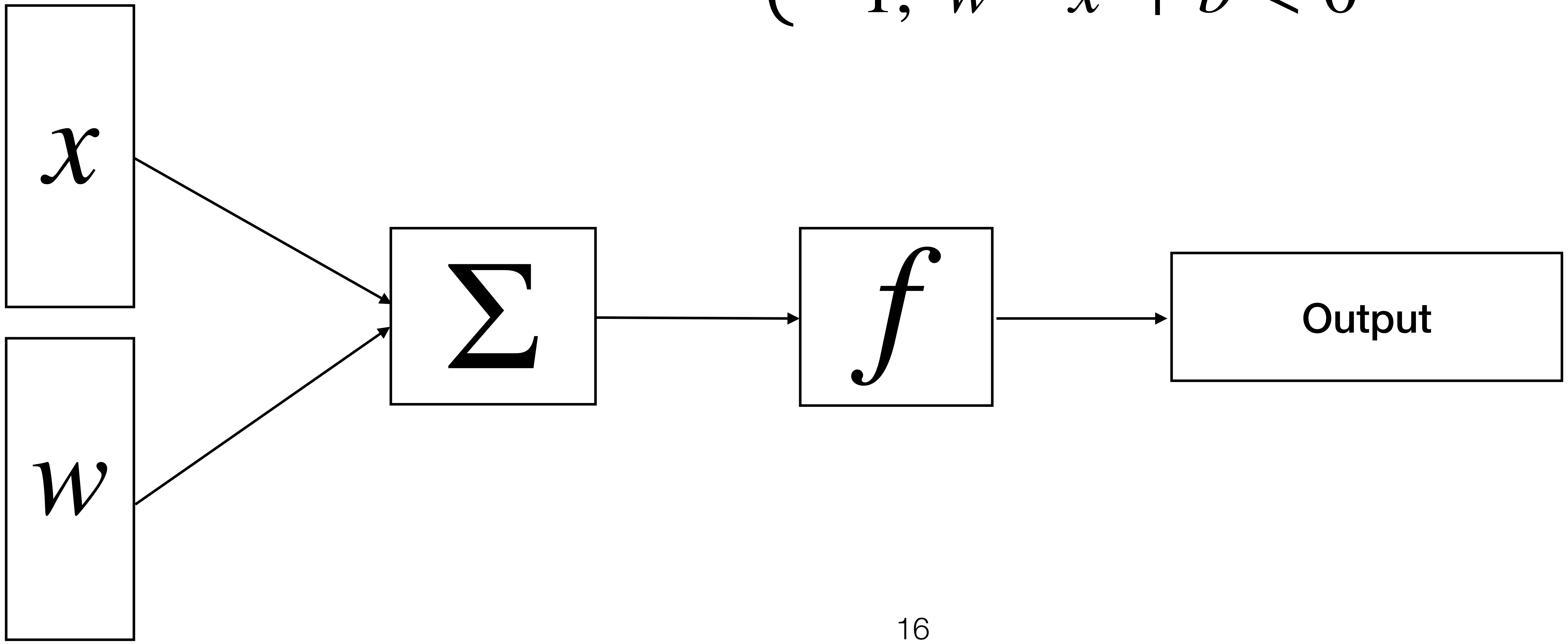
Supervised Learning II

- Typical setup for a supervised learning problem:
    - large dataset with known labels split between a training and validation set (90%/10%)
    - some kind of classification or regression algorithm
    - a non-ambiguous problem to solve
- Next we will talk about different algorithms for supervised learning

I. The Perceptron

The Perceptron I

- Based on biological model of a neuron (at the time of conception)
- Binary classification/regression only. Unable to solve non-linear problems.
- Mathematically simple: $y = \begin{cases} 1, & \vec{w} \cdot \vec{x} + b \geq 0 \\ -1, & \vec{w} \cdot \vec{x} + b < 0 \end{cases}$



The Perceptron II

- Trained with backpropagation and Stochastic Gradient Descent (SGD).

$$w_i(t) = w_i(t-1) + \eta \nabla_w L(w; x)$$

New, updated weight for current timestep

Weight from previous timestep

Learning Rate

Loss function

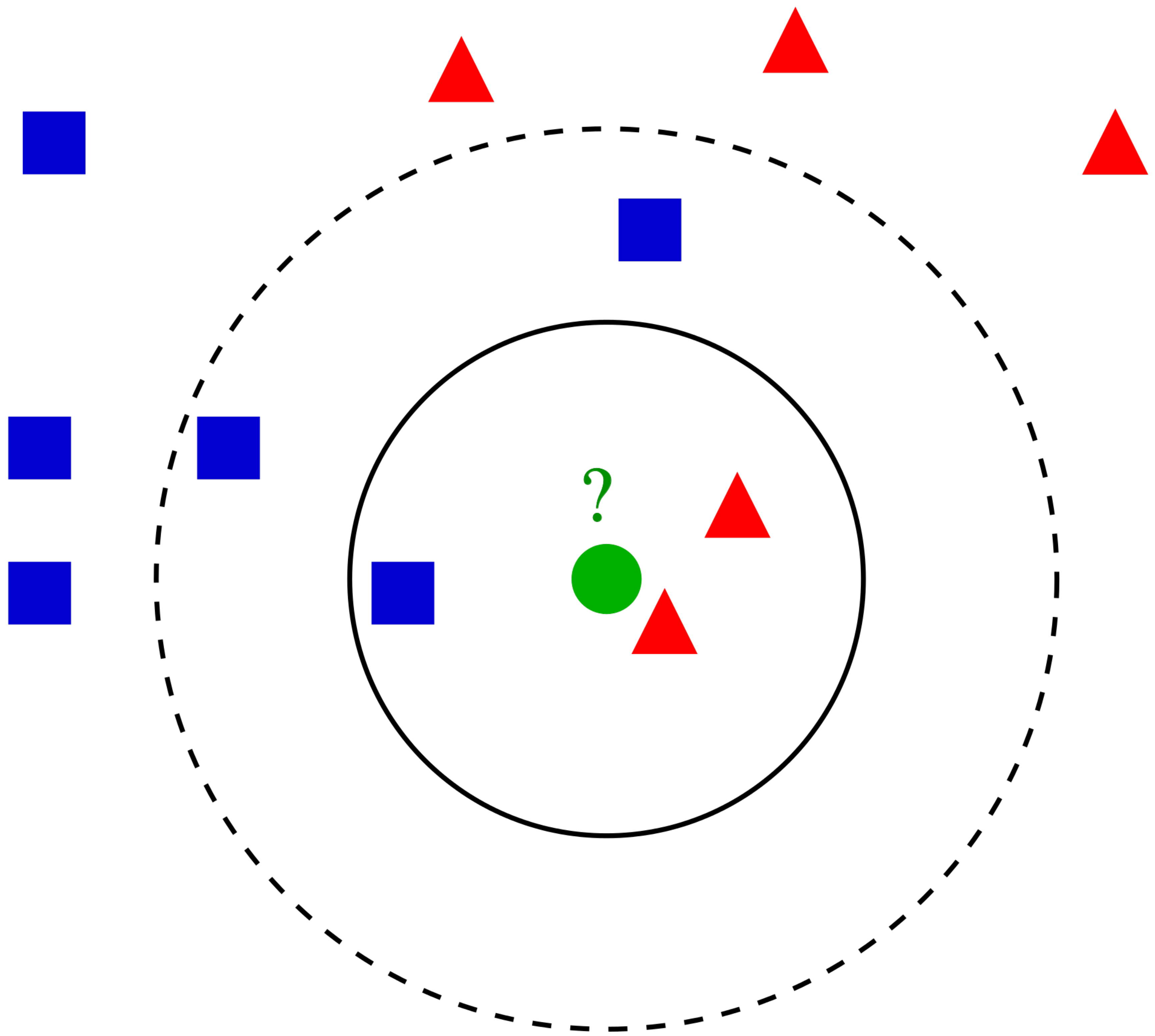
II. K-Nearest Neighbours (K-NN)

K-NN I

- For a point in your feature space, take the k nearest points to it and calculate a distance metric.
- In classification, a majority vote decides on the class label of the point
- In regression, an average of the distances decides where the point lies
- This implicitly calculates the decision boundary in your data

K-NN II

- k-NN is known as instance-based or lazy learning as it only ever approximates the function locally and it has no need to compute it globally



III. Gaussian Processes

Gaussian Processes I

- Each class in the dataset is distributed normally
- The prior of the process is then the joint distribution of the the classes in the problem i.e. our prior is a multivariate Gaussian
- We then want to use the covariance matrix between the points (often referred to as the kernel) to constrain the posterior distribution of our data

Gaussian Processes II

- More generally, we assume a Gaussian prior of a set of functions which may represent our problem and use the covariance of the data to constrain the posterior on this set of functions

$$p\left(y^* \,|\, x^*, f(x), x\right) = N\left(y^* \,|\, A, B\right)$$

$$A = K(\theta, x^*, x)\, K(\theta, x, x')^{-1}\, f(x)$$

$$B = K(\theta, x^*, x^*) - K(\theta, x^*, x)\, K(\theta, x, x')^{-1}\, K(\theta, x^*, x)^T$$

IV. Support Vector Machines (SVM)

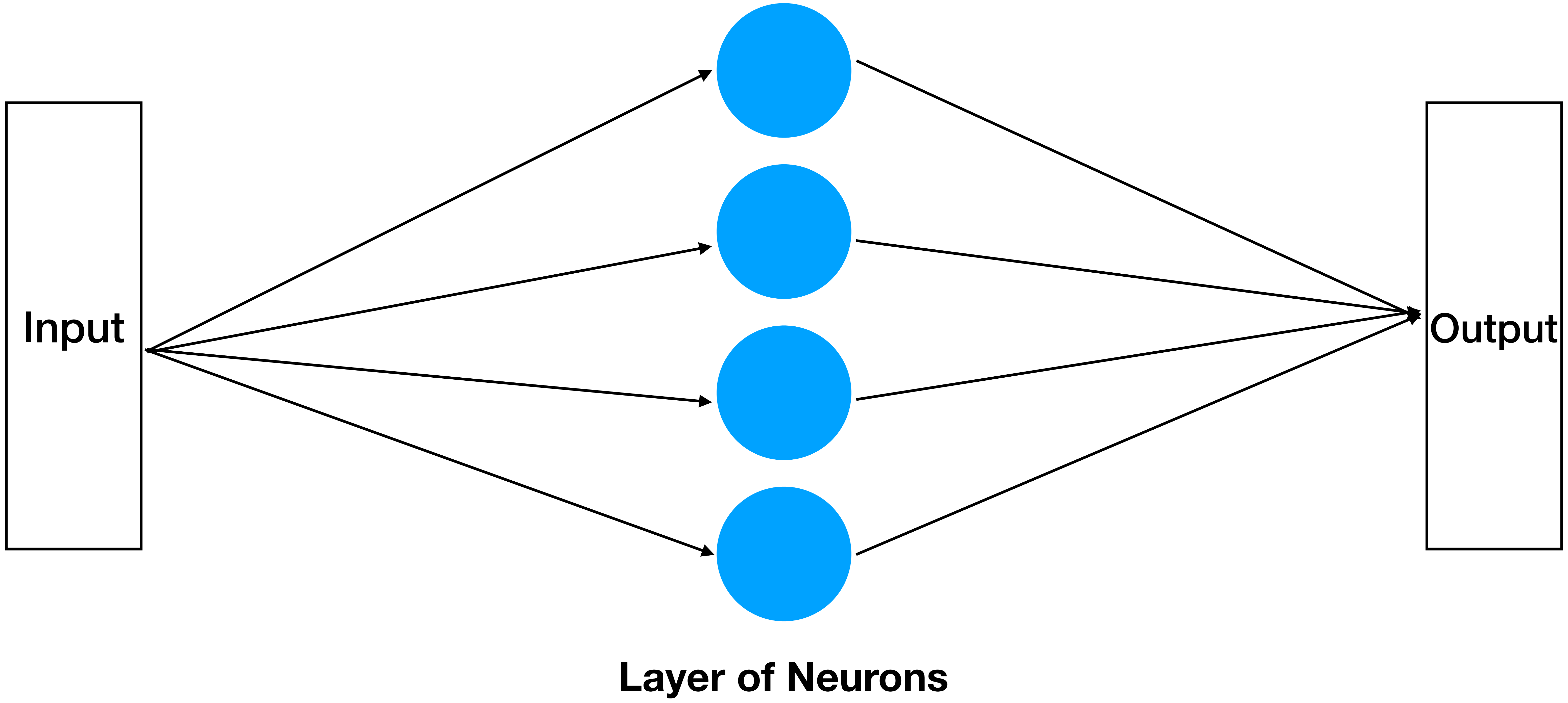
SVM

- Attempts to find a hyperplane which separates the classes in our dataset
- This is done via the use of “support vectors”
- For non-linear classification/regression tasks, SVMs find a non-linear transformation into hyperspace such that a linear decision boundary between the sets of data can be defined

V. Neural Networks (NNs)

- We will be specifically talking about shallow (single-layer) neural networks (for deep neural networks stick around until the last session)

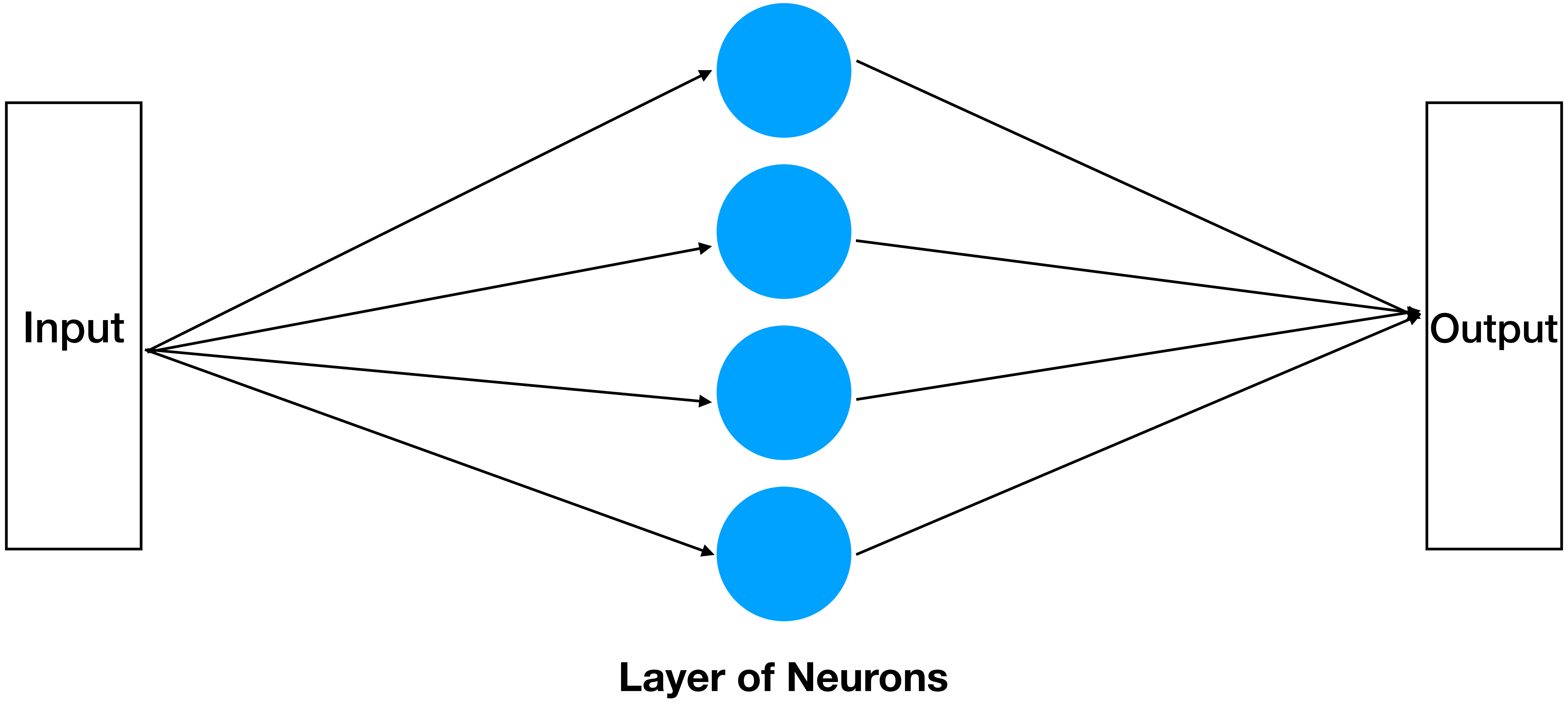
- These can be good for approximating functions and classification



NN II

- To train a neural network, we need to define a loss function and an optimiser (often SGD).

- NNs are difficult to train but can give incredible results when trained properly.



Tutorial

- Clone the repository at https://bit.ly/2D04zE1 if you haven't already
- Follow instructions to set up Python virtual environment
- We will be using 2 machine learning libraries: scikit-learn and PyTorch
- scikit-learn has most of the classical machine learning techniques built-in so no need to try and code your own
- PyTorch is a neural network equivalent of numpy and conversion between the two is easy: numpy array to Torch tensor is y = torch.from_numpy(x) and Torch tensor to numpy array is simply y = x.numpy().